CPSC 3300
Homework 1
Due 11:59PM Wednesday, January 29
Submit your answers to Canvas

Please provide sufficient space on your homework solutions so that your calculations and answers are easily readable and so that grading will be easier. Furthermore, except for the simplest questions, giving only the answer without showing your work is not acceptable. For the best chance at partial credit, show the generic equation you are starting with and any derivations needed to handle the information as given in the question, then plug in the values from the question. You may, of course, use a calculator for the homework. (Unlike the exams, the values in the homework questions are not necessarily chosen for ease of hand calculation.)

1. (30pt) A processor P has a 4.2 GHz clock rate and has a CPI of 1.5.

   (a) If the processor executes a program in 8 seconds, find the number of cycles and the number of instructions.

$$\begin{aligned} \text{Cycles} &= \text{Clock Rate} * \text{Time} \\ &= 4.2 * 10^9 * 8 \\ &= \mathbf{3.36 * 10^{10}} \end{aligned}$$

$$\begin{aligned} \text{Instructions} &= \text{Cycles} * \text{CPI} \\ &= 3.36 * 10^{10} * 1.5 \\ &= 5.04 * 10^{10} \end{aligned}$$

   (b) What is the MIPS rate for the processor?

$$\begin{aligned} \text{MIPS} &= \text{Clock Rate} / \text{CPI} * 10^6 \\ &= 4.2 * 10^9 \ / 1.5 * 10^6 \\ &= 4.2 * 10^9 \ / 1.5 * 10^6 \\ &= \mathbf{2.8 * 10^3} \end{aligned}$$

   (c) We are trying to reduce the execution time by 20% but this leads to an increase of 10% in the CPI. What clock rate should we have to get this time reduction?

$$\text{New Execution Time} = \%\Delta * \text{Execution Time}, \quad \text{New CPI} = \%\Delta * \text{CPI}$$

$$\begin{aligned} \text{Clock Rate} &= \text{Cycles} / \text{Time} \\ \text{New Clock Rate} &= \text{Clock Rate} * \%\Delta \text{ in CPI} / \%\Delta \text{ in Time} \\ &= 4.2\text{GHz} * 1.1 / 0.8 \\ &= 4.2\text{GHz} * 1.375 \\ &= \mathbf{5.775\ GHz} \end{aligned}$$

2. (20pt) Consider two different implementation of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (class A, B, C, and D). P1 has a clock rate of 2.5 GHz and CPIs of 1 (class A), 2 (class B), 3 (class C), and 5 (class D).

   Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 15% class A, 50% class B, 25% class C, and 10% class D.

$$\begin{aligned} \text{Class A} &= 1.5 * 10^5 \\ \text{Class B} &= 5.0 * 10^5 \\ \text{Class C} &= 2.5 * 10^5 \\ \text{Class D} &= 1.0 * 10^5 \end{aligned}$$

(a) What is the global CPI?

$$
\begin{aligned}
&\phantom{+}\ 1.5 * 10^5 * 1 \\
&+\ 5.0 * 10^5 * 2 \\
&+\ 2.5 * 10^5 * 3 \\
&+\ 1.0 * 10^5 * 5 \\
&= 2.4 * 10^6 \\
&/\ 10^6 \\
&= 2.4 \text{ CPI}
\end{aligned}
$$

(b) Find the clock cycles required to run the program on P1.

$$
\begin{aligned}
\text{Clock Cycles} &= \text{CPI} * \text{Instructions} \\
&= 2.4 * 10^6
\end{aligned}
$$

3. (20pt) Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 8, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 600 million arithmetic instructions, 80 million load/store instructions, 120 million branch instructions.

(a) Suppose we find a way to double the performance of the arithmetic instructions. What is the overall speedup of our machine?

$$
\begin{aligned}
\text{Old} &= (600 + 640 + 360) * 10^6 \\
&= (1600) * 10^6 \\
\text{New} &= (300 + 640 + 360) * 10^6 \\
&= (1300) * 10^6 \\
\text{Speed Up} &= \text{Old} / \text{New} \\
&= (1600) * 10^6 \\
&/ (1300) * 10^6 \\
&= 16 / 13 \\
&\approx 1.23
\end{aligned}
$$

(b) If we find a way to double the performance of the load/store instructions, what is the overall speedup of our machine?

$$
\begin{aligned}
\text{Old} &= (600 + 640 + 360) * 10^6 \\
&= (1600) * 10^6 \\
\text{New} &= (600 + 320 + 360) * 10^6 \\
&= (1280) * 10^6 \\
\text{Speed Up} &= \text{Old} / \text{New} \\
&= (1600) * 10^6 \\
&/ (1280) * 10^6 \\
&= 5 / 4 \\
&= 1.25
\end{aligned}
$$

4. (30pt) On machine newton using the **perf** tool, examine how compiler optimization levels and options change the number of instructions for the program **whetstone** and the number of CPU cycles to execute the program. Use gcc to compile your program. Refer to the following web page for information on how to use **perf** to count the number of instructions and cycles among other statistics: https://perf.wiki.kernel.org/index.php/Tutorial#Counting_with_perf_stat

I: Download the **whetstone** benchmark to your home directory:
http://www.netlib.org/benchmark/whetstone.c

Compile whetstone. You may need to explicitly specify the math lib folder and link to it, e.g.,

```
gcc -o whetstone whetstone.c -lm
#link the math with -lm
```

II: Examine the performance of **whetstone** looping 200,000 times (**./whetstone 200000**) compiled with the following levels/options:

a. -O0
b. -O1
c. -O2
d. -O3
e. -O3 -funroll-loops

Use a table to show the instruction count, #cycles, IPC, and time for each of the experiments, and calculate the speedup based on the execution time with -O0. Paste your screen shot at the end.

| | IC | #Cycles | IPC | Time (ms) | Speedup |
|---|---|---|---|---|---|
| -O0 | 22,781,415,296 | 17,412,450,517 | 1.31 | 6816 | 1 |
| -O1 | 10,970,050,245 | 8,912,462,018 | 1.23 | 3654 | 1.865 |
| -O2 | 6,048,468,884 | 6,151,232,967 | 0.98 | 2380 | 2.863 |
| -O3 | 6,021,566,219 | 6,113,609,038 | 0.98 | 2372 | 2.874 |
| -O3 -funroll-loops | 5,008,169,917 | 5,443,910,528 | 0.92 | 2097 | 3.250 |